ARCHITECTURE OF A COMMON MILITARY SCENARIO DEVELOPMENT ENVIRONMENT FOR SIMULATION

ABSTRACT: *Functionality of a Military Scenario Development Environment (MSDE) is traditionally implemented in a proprietary fashion. Virtually every simulation system in existence today utilizes a specific exercise file format that is unique to that system. Each time a new simulation system has been developed, scenarios have been created in proprietary formats to support exercises in that simulation. All too often, these exercises are re-created by re-keying data out of other existing simulation exercise files or training packages. Such redevelopment of exercises is time consuming and costly. Frequently, simulations share similar architectures, and even training objectives. Each utilizes data and data structures that model real world weapon systems, organizations, time, and environment. A great number of standards are available that support modeling of complex systems. Relational database standards define rules of referential integrity for data. Extensible Markup Language (XML) provides for data type definitions in portable ASCII format. Microsoft Office provides standard desktop interfaces typically used by customers and users of military simulations for planning and after action review (AAR). Other standards exist or are developing that support integration of simulation with native C4I systems. This paper describes how a common MSDE software capability could provide for interoperability of simulations across disparate platforms in a fashion that integrates the desktop applications and C4I systems already in common use by the simulation end user.*

Jeffrey B. Abbott
AcuSoft Inc.
Orlando, Florida

Mr. Jeff Abbott is a Project Manager for AcuSoft, Inc. He holds a Masters Degree from the University of Central Florida in Electrical Engineering and Communications Systems. He has 16 years experience in the development and management of software systems for government contracts. He is currently working on Military Scenario Development and AAR applications based in Microsoft Office.

Doug Parsons
AMSTI-EC STRICOM
Orlando, Florida

Mr. Doug Parsons is a Sr. Systems Engineer at STRICOM's Intelligent Simulations Division. He holds a Masters Degree in Systems Management (Operations Research) from the Florida Institute of Technology and is currently working toward a second Masters Degree in Industrial Engineering (Interactive Simulation and Training) from the University of Central Florida. His technical interests include software systems test design, software reliability, and cognitive modeling.

Jesse Liu
AcuSoft Inc.
Orlando, Florida

Mr. Jesse Liu, President of AcuSoft, Inc., holds a Masters Degree in Electrical Engineering from the University of Florida. Mr. Liu has simulation work experience for General Electric, Sun Microsystems and start-up software companies. Mr. Liu has 16 years experience in simulation.

Bud Dannemiller
Litton PRC
Fort Knox Kentucky

M.A. (Bud) Dannemiller is a Senior Computer Systems Analyst with the field office of Litton-PRC (a Northrop-Grumman company) at Fort Knox, KY, and holds a Bachelor of Arts Degree from Loyola University, New Orleans. He completed 10 years of active federal service with the U.S. Army and has spent the last 13 years as a government contractor focusing on developing solutions for simulation systems and structured, simulation-based training.

ARCHITECTURE OF A COMMON MILITARY SCENARIO DEVELOPMENT ENVIRONMENT FOR SIMULATION

Jeffrey B. Abbott
AcuSoft Inc.
Orlando, Florida

Doug Parsons
AMSTI-EC STRICOM
Orlando, Florida

Jesse Liu
AcuSoft Inc.
Orlando, Florida

Bud Dannemiller
Litton PRC
Fort Knox Kentucky

INTRODUCTION

The integration of open and interoperable technologies with new and existing modeling and simulation (M&S) architectures shows great promise to the industry as a whole. Under OneSAF and other programs, the simulation and modeling industry is applying these technologies to integrate simulation, planning, and operational applications through the sharing of scenario and model data. Many commercial standards are available to enable this integration through various standards of interoperability. These standards have evolved over the years. Some, such as XML, have evolved at a frantic pace, while others, such as open database connectivity (ODBC), have ebbed slowly forward over the decades. These technologies not only show promise for interoperability in simulations, but also represent the promise of interoperability of simulations with desktop applications in common use by simulation users. This integration can produce familiar operating environments from which the user can apply simulation to their training and research needs. Interoperability will take on a whole new meaning when the M&S community user no longer views simulation as a disparate technology base built upon proprietary formats and applications. Instead, simulation will become a technology that is an extension of the individual that works the way they do. The benefits of such integration cannot be measured, only imagined. This paper will present a model for interoperability within a MSDE that lays the groundwork for M&S developers who need to know how it is done.

Background

The role of a military scenario development environment is to describe and define an exercise utilizing simulation, and define how that exercise is to operate. The description and definition include:

- Battlefield graphic control measures and overlays
- Execution plans
- Task organization lay down (units, equipment, platforms, etc.)
- C4I utilization details
- Environment and weather

Today, the functionality of a MSDE exists in technologies supplied by various component applications utilized in simulation. These components cover the entire operational cycle of plan, prepare, execute, consolidate, and reorganize. But without the integration required to allow these technologies to interoperate, a complete model of a MSDE that supports this cycle has yet to be realized. This paper will present technologies that may be used to provide this level of interoperability, and will describe an environment that models the operational military scenario development process in use today. Interoperability is discussed in the context of data interchange. The interchange or sharing of descriptive information is distinguished from the exchange of application level data. In this paper, data that is exchanged between applications (back and forth) is referred to as *data exchange*. Data that is shared from one application to another (one way), in an information only context, is referred to as *information sharing*.

Classes of applications
The classes of applications that need to exchange scenario related data include:

- Simulation applications such as ModSAF, Janus, WarSIM, OneSAF, and others
- Modeling applications used to define the entities and simulation objects in general
- Desktop (training & planning) applications that support training support products (TSPs), and after action review
- Operational applications that integrate with simulation such as Advanced Battle Command Systems (ABCS) C4I equipment, forward entry devices (FED), global positioning satellite equipment, etc.
- MSDE that utilizes aspects of all these applications in a manner that supports the composition of scenarios utilizing each application's data and technology

Requirements for interoperability
For data to provide the mechanisms for interoperability in simulation, there needs to be a method for applications to validate, interpret, and understand the language in which the data is represented. In the past, the language used has been controlled by the applications that empower the simulation. The M&S community needs an extensible language capable of implementing vocabularies that organize data from the perspective of the applications in which the data is to be used, but at the same time the language needs to be data centric, i.e. represent the data from the perspective of the data itself.

Information and data to be shared
The data that needs to be shared and/or exchanged between these applications prior to, during and after exercises utilizing simulation include:

- Task organization data to provide a common view of the units participating in or being simulated within the exercise.
- Environment and environmental conditions. Interchange of environmental data ensures each application represents and models the environment in a consistent manner.
- Graphical control measure data to provide common points of reference on the battlefield.
- Execution and synchronization matrixes. This data organizes behaviors for computer generated forces (CGF) units and missions/orders/collective tasks for operational units in a manner that enables synchronization of the simulation battlefield.
- Communications network data to allow simulated and operational systems to interoperate. The utilization of native C4I systems allows the user to coordinate simulation exercises from their own equipment.
- Application identification data of simulation related objects. This data links simulation objects to specific applications. For example, which simulated/operational unit does a particular Force XXI Battle Command Brigade and Below (FBCB2) unit reference number (URN) identify?

The benefits of exchanging such data promise to "Reduce the time and cost to develop interoperable scenario-related tools; Improve the quality of scenario-related tools (Lacy; Dugone, 2001 p. 2).

Method of Data Exchange
The primary method of data exchange is the language used to describe the data. A data exchange language must:

- Be data centric to ensure it will not change over time with the requirements of any given application.
- Provide a clear organization of data that supports portability across heterogeneous applications by defining the structure, constructs, and content in a manner that can be meaningfully interpreted by these applications.
- Provide mechanisms for translating a common scenario file vocabulary into application-specific vocabularies and even application-proprietary formats.

Benefits To Be Realized
Information need only be defined/entered once. It can then be shared or exchanged with other applications that need to understand, reference, or operate against the data. For example, FBCB2 sends messages between units identified by unit reference numbers (URN). If the URNs and their related unit designations/descriptions are not understood by other applications, the messages mean little. If this data is not shared with other applications, interoperability may break down, as FBCB2 URNs change. Through the sharing of FBCB2 data, other applications can understand FBCB2 vocabulary and represent FBCB2 messages in meaningful ways. Whether simulation exercises are applied to support either training or research objectives, the sharing of these objectives with other applications will focus the intent of the exercise. For example, a simulation operator will more readily recognize when the exercise is off track if he has knowledge of the training objectives of an exercise. Through interoperability, operational applications can be reused within simulation. This means that the simulation participants can utilize exactly the same systems as they do in live

environments, reducing negative training and simplifying research development and analysis tests and testing environments.

BASIC APPROACH

Simply put, the basic approach to providing for interoperability of simulation data is to identify the architecture for application interoperability, the mechanisms by which data is exchanged within the architecture, and the available mechanisms for integrating applications into the architecture.

Architecture of Interoperability

While there is a great deal of information available concerning mechanisms of interoperability in terms of language, constructs, and semantics, very few interoperability initiatives describe the architectures that empower interoperability among applications. A standout exception is the Synthetic Environment Data Representation and Interchange Specification (SEDRIS). The SEDRIS web site identifies SEDRIS as "(1) representation of environmental data, and (2) the interchange of environmental data sets." The web site further describes the initiative:

"As an interchange mechanism, SEDRIS is at the crossroads of many diverse IT applications that require environmental data. And its powerful representational concepts and schema have already influenced many authoring, database generation, and IT applications … the fundamental aspects of SEDRIS remain the unambiguous representation of environmental data and the efficient interchange of such data through its software technologies".

The interchange of scenario data can provide similar benefits to simulation interoperability as SEDRIS does for the synthetic environment. The SEDRIS architecture is composed of five technology components consisting of the SEDRIS Data Representation Model (SDRM), Environmental Data Coding Specification (EDCS), Spatial Reference Model (SRM), SEDRIS interface specification or Application Programming Interface (API), and the SEDRIS Transmittal Format (STF). The first three components (SDRM, EDCE, and SRM) are used as the mechanism for describing environmental data. The last two components (SEDRIS API and STF) represent the technologies that empower SEDRIS to share and exchange environmental data in an efficient manner. The overall architecture for SEDRIS and SEDRIS interoperable applications is represented in Figure 1.

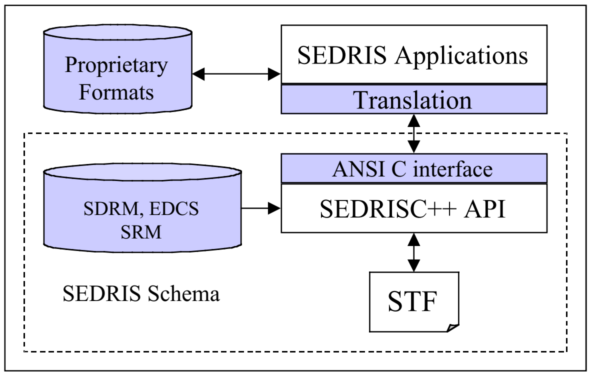


Figure 1. SEDRIS Architecture

This architecture lends itself well as a starting point for MSDE and scenario data interchange across simulation. As noted earlier, MSDE is itself an application. The mission of MSDE is to provide for interoperability across applications by way of data interchange. As an application, a MSDE needs to present scenario data in a structured and familiar fashion to the user. A MSDE needs to interact with the user to define the scenarios that integrate data from each application within the simulation environment.

Mechanism of Data Exchange (XML)

As with SEDRIS, modeling & simulation needs a mechanism for data interchange. XML and its related technologies provide such a mechanism. The promise of XML has not gone unnoticed by the M&S community. As an initiative under OneSAF, XML has been selected as the language of choice for describing data (scenarios, models, messages, etc). Under the High Level Architecture (HLA) Dynamic Scenario Builder (DSB) initiative, XML has been selected as the data interchange format for representing and exchanging scenario data. XML and SEDRIS share architectural similarities. XML as a markup language provides the unambiguous semantics within which a particular vocabulary's structure and content may be defined, represented, and transformed to other vocabularies.

Data Type Definition Language (DTD)

A single XML based vocabulary is specified by a data type definition. DTDs have been described as "a language for formally declaring the vocabularies we use. This allows our tools to constrain the creation of an instance of our information, and allows our users to validate a properly created instance of information against our set of constraints" (G. Ken Holman, 2000, p. 2). DTDs can be used in simulation for the same purposes to allow a scenario or model developer to validate this data prior to executing it in simulation. DTDs support subsets as a means by which XML vocabularies may be scoped within a file that is shared across vocabularies. Each subset can be applied to the application for which the data is intended.

Dynamic Object Model (DOM) API
The DOM API provides methods for representing and manipulating XML document structure and content. The DOM API may be used to access and build XML datasets as defined in their DTDs. The DOM specification is being extended to support events that notify an application when a particular element of an XML file has been reached. Events will allow an XML application's code to perform application specific functions in response to data, such as triggering the display of a form to specify the control measures and other attributes needed to execute a unit behavior.

Extensible Stylesheet Language (XSL)
The extensible stylesheet language (an XML draft) provides a means for specifying how an XML vocabulary is to appear to the user. It is a language that specifies the presentation of an XML vocabulary. Using XSL, scenario files can be presented in different manners depending upon the needs of the user. For example, an Army user may want to view a scenario file in the form of a Training Support Package. Another user may want to explore the scenario as a set of objects and properties, providing easy access to weapon system models, unit behavioral models, etc.

Extensible Stylesheet Language Transformations (XSLT)
Another language of XML is the Extensible Stylesheet Language Transformations or XSLT. XSLT is used as a language for transforming a document (an exercise file for example) from one vocabulary to another. XSLT was designed to transform XML vocabularies when the data is not structured for applications other than that for which it was created. In his paper entitled What is XSLT?, Ken Holman states "XSLT 1.0 recommendation describes a vocabulary to transform information from an organization in the source file into a different organization suitable for continued downstream processing." Clearly, XSLT is an ideal mechanism for exchanging simulation data between applications. For example, XSLT could be used to transform a scenario file into HTML for viewing over the Web or even within Microsoft Office products such as Word, Excel, or PowerPoint.

Resource Description Framework (RDF)
RDF provides a method of describing and interchanging metadata. RDF provides a convenient method of describing an XML file. It can be utilized to produce a thumbnail summary of a document to feed a card catalog system for searching a library of documents. For simulation & modeling, the RDF could be used to feed catalog systems of scenarios, models, or training support packages. RDF can identify the audience for which a scenario was developed, the objectives of the scenario, a summary of the scenario's initial conditions, assumptions, sequence of execution, etc. In short, it can be used to cross-reference a scenario by whatever information is applicable to the user. RDF provides a mechanism for scenario and simulation model retrieval.

Mechanisms of Application Integration

The mechanisms and technologies for simulation data exchange and sharing have been covered. However, even if an infrastructure utilizing these mechanisms existed today, how would developers integrate their applications with that infrastructure? The approach is simpler than it might seem.

Interface Specifications
Today there are a number of specifications that define the "bindings" or interfaces that allow code written in one language to be reused in other languages under heterogeneous operating environments. The Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), SUN Micro Systems' Java Native Interface (JNI), and Microsoft's Component Object Model (COM) and Distributed Component Object Model (DCOM) specifications provide architectural standards that allow applications to interoperate, or interact, in a well defined (object-oriented) fashion. XML API product vendors are supporting Java bindings, OMG IDL specifications as defined in COBRA 2.3.1, and Microsoft provides COM and DCOM XML products as well as COM/DCOM development environments. COM and JNI are of particular interest to MSDE. These standards are binary compatible, which means they use the same calling convention. Reusable components written to support the CORBA, COM and JNI specifications have distinct advantages over the use of classical external components or libraries. Unlike classical external libraries, the interface of reusable components does not need to be declared in the language of the application in which it is used. These interface specifications allow the developer of reusable components to easily place error-handling code within the interface itself. Even the properties

(variables) exposed through the interface may include code to verify or otherwise operate on the inputs before they are used. This enables the developer to fully encapsulate the internal behavior of objects within the reusable components they are redeveloping. The external interface can be sufficiently abstracted from the internal operations of a component so that dependencies of an application on a particular component's implementation can be all but eliminated.

Object Component Architecture

In instances where the use of these specifications does not fit the tasks of interoperability at hand, software interoperability may be achieved through the development of interfaces designed to support reliable interactions between applications. For example, if a set of C++ classes needs to be shared between existing applications that have been written in different languages, are to be executed on different operating systems, and/or are to be run on different hardware architectures, an interface that supports all aspects of such a heterogeneous architecture must be developed. The following figure depicts an example of a set of C++ libraries with appropriate interfaces that allow them to be accessed or reused by an Ada application.

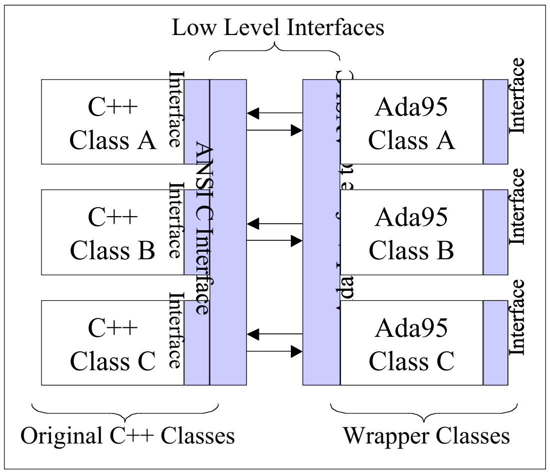


Figure 2. Component Interface Example

The first step is to develop wrapper classes, in Ada, that mimic the original object-oriented structure of the underlying C++ classes. These wrapper classes provide Ada applications with a native Ada interface to the C++ objects. Next the low level interfaces between the C++ classes and the Ada wrapper classes need to be developed.

The low level interfaces flatten out the object model services, effectively merging all class services under two interfaces. One interface exports the class services from C++. The second interface imports the class services to Ada. Each of these low level interfaces is defined to use the same calling convention, preferably the ANSI C standard. The result is an unambiguous set of Ada classes and associated services that match those of the original C++ classes. At this point, interface has been completed. However, in order to make the C++ classes reliable components for reuse, additional work needs to be done to insulate Ada from C++ and C++ from Ada. If an unhandled exception occurs in the C++ side, it can adversely affect the Ada application utilizing the C++ components. The approach is to write code within the exported interface layer that is designed to handle all errors when they occur. Although a reliable interface can be created in this fashion, standardized interfaces such as CORBA and COM should be used when ever possible.

Integration Languages

Two of the more popular languages that provide for interoperability between applications are Java and Visual Basic (VB). Both Java and Visual Basic support object models designed for reuse and interoperability across applications. When applicable, the use of these languages can have a significant impact on maintainability and reliability of software systems. New and existing software components can be encapsulated (wrapped) with reliable interfaces that allow VB and Java to easily integrate the components into a "whole" application. Figure 3 illustrates this concept graphically.

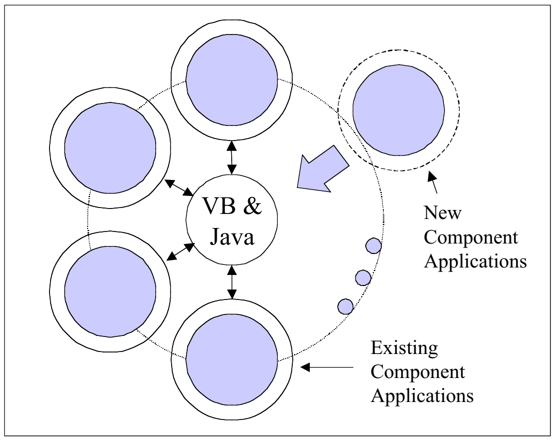


Figure 3. Integration Languages

It should be noted that no single language provides an "end all" solution to all software problems. When applied improperly, the strengths of any language can become a liability. It is important that the developer choose the right tool for the right job.

Java and the Java Native Interface (JNI)

Java has the strength of being itself interoperable across different operating systems and hardware platforms. Many venders support Java bindings that allow the language to interface with external software components. In their discussions on the overview of the JNI, Sun Microsystems states:

"By programming through the JNI, you can use native methods to:

- *Create, inspect, and update Java objects (including arrays and strings)*
- *Call Java methods*
- *Catch and throw exception*
- *Load classes and obtain class information*
- *Perform runtime type checking*

You can also use the JNI with the Invocation API to enable an arbitrary native application to embed the Java VM. This allows programmers to easily make their existing applications Java-enabled without having to link with the VM source code."

A shortcoming of the JNI is that its implementation is specific to the vendor providing the Java Virtual Machine being used. However, since applications that are to make use of the JNI are generally reside specific platforms, this is not as large a drawback as it might seem. In any case, a JNI provides the capability for Java to interact with applications that have exported their objects and services through the JNI, making it a very powerful integration tool when applied to interoperability among applications.

Visual Basic and Component Object Model (COM)

Microsoft's COM interfaces can be easily created and used within Visual Basic 6.0 (VB6). COM interfaces are particularly interesting as the interface exposes itself to Visual Basic, just as the JNI does for Java. This allows a compiler to export the interface when a COM component is built, and import it when a COM component is used within another application. As with JNI, applications can expose themselves to VB by implementing a COM interface. Microsoft has gone to great extents to make COM a technology designed for reuse and interoperability across its products and languages. For example, Microsoft Office products are all COM capable. Not only can they access any COM object, they are themselves COM objects that are accessible from Visual Basic. In addition, office applications are capable of executing VB code directly by including Visual Basic for Applications (VBA) support. VBA is analogous to using JNI with the Invocation API. When an individual records a "macro" in an office application, they are actually recording the COM calls being made to the application within the structure of VB. Because any VB or "VBA enabled" application can include references to any COM compliant application, Visual Basic is able to coordinate the execution of tasks across applications such as Word, Excel, Outlook, and PowerPoint with any other COM object

It should be noted that the Microsoft Raw Native Interface (RNI) for Java is capable of making COM calls and of defining COM interfaces to Java components. Java running under the Microsoft Java VM can be used as easily as VB/VBA to integrate COM application under windows. However, cross platform support for COM is not yet available.

Comparisons of the Technologies

Clearly Java and Visual Basic are tightly integrated with the interface specifications they support. Without this integration, neither would be of particular interest to interoperability. Both are capable of being used as part of another application, or of being used as an application that utilizes other applications as software components. It is the differences in operational environments and interface technologies supported by Java and Visual basic that is key to the application of each.

Visual Basic has the advantage of being integrated directly with Microsoft Office applications. Microsoft Office's huge desktop user base and general familiarity of the office interface makes Visual Basic an ideal tool for interoperability within the Windows environment. Visual Basic can easily be applied as an integration language to enable the interoperability of Microsoft Office with simulation data, data type definitions, and even the simulation applications themselves all from the desktop applications with which the user is already familiar.

Java has the advantage of cross-platform support. Any system on which a Java Virtual Machine is available can execute Java code. Because Internet browsers such as Netscape exist with Java VM support on nearly all platforms, Java offers native interoperability support across hardware platforms. Java inherently supports JNI, but only supports COM under the Microsoft Java VM. However, as indicated by Sun Microsystems (1997), because JNI and COM are binary compatible, COM support is expected for Java once COM support becomes available to non-windows operating environments.

THE MSDE MODEL FOR SIMULATION

The proceeding technical discussion provides the foundation for presenting a MSDE that is capable of modeling the way today's war fighter develops scenarios or plans for actual military operations (battles). A MSDE should guide the user through the scenario development process in the same manner that actual battles are planned. In short, a MSDE should model the way the Army develops actual battles just as simulation models the execution of each engagement phase of the battle.

Modeling the Plan

To model the military scenario development process, it is necessary to describe the details of the process and the dynamics used by the Army to develop plans. Operations of battles utilize an adaptive process of Plan, Prepare, Execute, Consolidate, and Reorganize. This represents a cyclical process that is executed to allow commanders to continually adapt the plan to the battle as the situations change. Because this process is cyclical, it is important that a MSDE support the process in a cyclical fashion as well. At each step in the process the commander utilizes three principle methods that provide structure to the process. First the situation is analyzed. Second resources are integrated, coordinated, and prepared. Third, resources are synchronized to carry out the plan quickly and decisively.

Analyzing the Situation.

The planning phase of a battle does not end when the execution, or simulation, begins. The plan is continually revisited and re-assessed based on the applicable mission, enemy, troops, terrain and weather, and time available (METT-T) that relates to the battlefield. In Field Manual 100-5 Operations, The Department of the Army gives METT-T the following context:

"The attainment of intermediate objectives must directly, quickly, and economically contribute to the operation. Using the analytical framework of mission, enemy, troops, terrain, and time available (METT-T), commanders designate physical objectives ... essential to accomplishing the mission."

The factors of METT-T are used to estimate the situation. These estimates are applied to decisions made at each step of a military operation as the operation cycles from phase to phase.

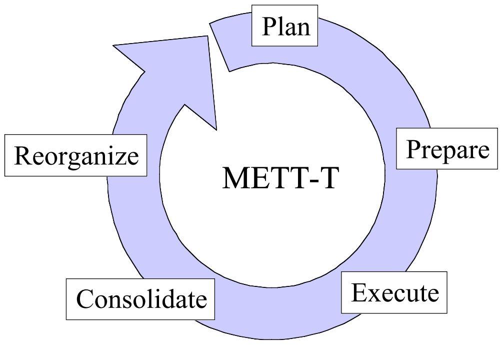


Figure 4. Cyclical Process of Operations

Integrating, Consolidating, and Preparing

In order to accomplish the mission, METT-T is used to feed the functions that provide superiority on the battlefield. These functions are referred to as Battlefield Operating Systems (BOS) and are used to examine the situation. Battlefield Operating Systems include:

- Intelligence (INTEL) is the collection and analysis of information.
- Maneuver (MAN) is the movement of resources relative to the enemy.
- Fire Support (FS) is the collective and coordinated employment of fires.
- Air Defense (AD) is the defense or protection against enemy air attack.
- Mobility, Counter Mobility and Survivability (MCM) is the support of mobility of forces and the denial of mobility to the enemy.
- Command Control and Communications (C3) is the visualization, direction, control, and communications of the intent of operations.

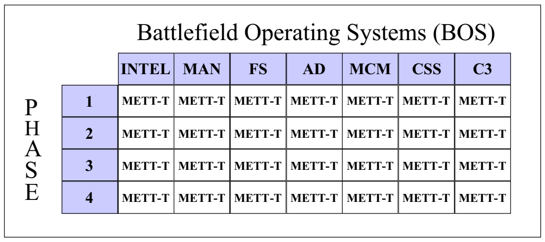
For tactical operations, the Department of the Army states that Battlefield Operating Systems:

"Enable a comprehensive examination in a straightforward manner that facilitates the integration, coordination, preparation, and execution of successful combined-arms operations."

These functions exist at all levels of war (tactical, operational, and strategic), and are used to focus tactical estimates of METT-T factors on the specific systems that operate on the battlefield. An Army commander synchronizes these functions to build and sustain superiority on the battlefield.

Synchronizing the Plan

Synchronization of the plan ensures the desired effect is achieved through the arrangement of BOS in time and space. When presented in context to the phases of a battle, METT-T, BOS, and synchronization may be visualized as a matrix in which engagement phases represent a time-line, BOS represent the functions that must be synchronized at each phase (point in time), and METT-T provides the basis for analyzing the tactical application of these functions at each phase of the battle. This matrix is illustrated in the following figure.



Battlefield Operating Systems (BOS)

PHASE	INTEL	MAN	FS	AD	MCM	CSS	C3
1	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T
2	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T
3	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T
4	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T	METT-T

Figure 5. Synchronization Matrix

In military battles, each phase follows the sequence of Plan, Prepare, Execute, Consolidate, and Reorganize. Historically, the significance of the "consolidate" and "reorganize" phases has not been well understood or modeled in simulation. For example, a battle may involve four principle engagement phases consisting of Movement to Contact, Defend, Withdraw, and Counter Attack. During an actual operation, a military organization consolidates its resources following each operational phase to reassess their available strength and resources. Following this assessment, the organization is "reorganized" to optimize its strengths and minimize its weaknesses. Following each "reorganize" phase, the plan is revisited. The plan is adapted to fit the actual situation represented by the METT-T of the battlefield. This situation is nearly always different from what had been anticipated prior to the execution of each operational phase. The commander must adapt his plan to reflect actual strengths or weaknesses that had not been anticipated. A MSDE, as a planning tool, must interact with the simulation to allow the situation to change, in an expected or unexpected manner, from one phase of a battle to another. A MSDE needs to support the analysis of METT-T factors for situational assessment of battlefield operating systems and integrate these in a fashion that enables synchronization of the plan in a clear, uncomplicated, and concise manner.

The MDSE Model

An accurate MSDE model needs to support the planning of a scenario for each phase, or engagement, making up a military battle. Each phase should be planned making assumptions concerning the desired METT-T conditions that are to exist following one phase and leading into the next. For instance, in the example already cited, strength of forces available for the Defend phase depends directly on the readiness of the forces as they exit the Movement to Contact phase. Each phase of such a battle would typically be run as an exercise in and of itself. However, if the ending conditions (force lay down, damage, readiness, etc.) of one engagement are not fed into the next, then the commander will not have to adapt his plan to the situation. Training effectiveness of the simulation environment with respect to the commander falls short. A MSDE should be held to the same standard of realism as the simulation models of execution. An architecture that provides such adaptive planning capabilities can be developed though the incorporation of a synchronization matrix with situational feedback from one phase to another, as illustrated in the following figure.

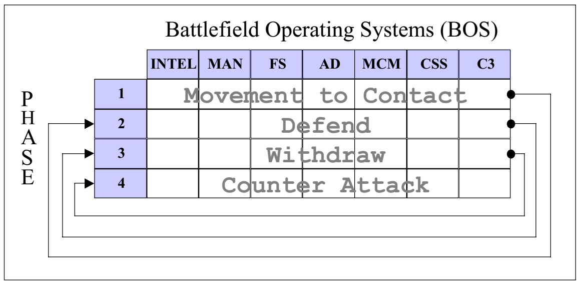


Figure 6. Synchronization Matrix Example

A MSDE can use a synchronization matrix interface to allow the scenario planner to specify (map) units and unit orders (behaviors) for each cell of the matrix. The synchronization matrix interface can be used to integrate multiple engagement scenario files, as phases, into an overall battle (set of exercises), as well as organize the phases of the battle within a single scenario file. The synchronization matrix would not necessarily provide any information directly to the simulation, but would provide the necessary organizational model of the scenario's "execution" to the user. By including METT-T analytical functions within the scenario planning tools of a MSDE, commanders can analyze the situation at each phase (scenario) of a battle. The key to effective METT-T integration within a MSDE is to make the correct analytical tools and data available at the correct time. Interoperability with simulation assets and interchange of scenario/model data hold the key that will enable the use of accurate METT-T analytical tools and adaptive planning within a MSDE.

Interoperability and the MSDE Model

The technologies that enable interoperability can be integrated with simulation to implement a MSDE model that supports the development of conventional views of operational plans (OPORD, Execution Matrix, Overlay, etc), provides METT-T situational analysis tools, clearly synchronizes the plan within the scenario development process, and works seamlessly with operational and simulation applications. The following figure depicts this concept.

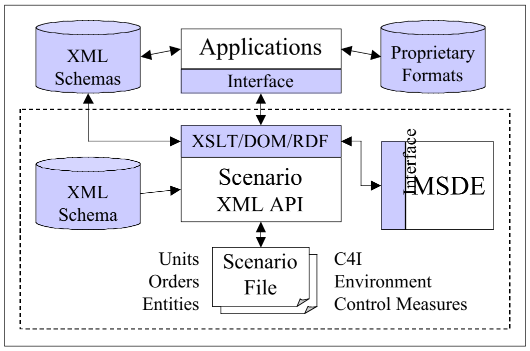


Figure 7. Sample MSDE Architecture

Operational Views of Scenario Data
Operational views of scenario data may be achieved through the use of XML schemas in combination with operational/training data and the respective applications. These schemas can serve as templates designed to fill portions of training support packages. XSLT can be used to merge the details of a scenario file with the schemas and to generate the organizational context of operations orders, execution matrixes, observer controller materials, training and evaluation outlines, maps and overlays, etc. Because Microsoft Office supports XML, the resulting data can be produced in the vocabulary of the Office application for which it is intended. RDF can be utilized to catalog and support search and retrieval functions of a scenario library within the user's desktop or over the web.

Operational Process
Data exchange can be used as part of an adaptive planning process that forces commanders to adapt their plan to changes on the battlefield. As each phase of a battle is completed in the course of an exercise, the simulation can exchange scenario data with the MSDE. This scenario data would represent the exit state of one phase of the battle that is to be fed into the next. A MSDE can use this data as the s*tate of the battlefield* that is applied to the next phase (scenario file) of the battle. The commander can adapt his plan to the new situation as represented in the modified scenario. The scenario can then be saved to disk, loaded into the simulation, and used to execute the next phase of the operation. This process can be extended to other assets within the simulation environment. For example, AAR data collected in the course of one exercise can be used to modify and/or augment OPORD (mission, concept of operation, commander's intent, enemy situation, friendly situation) for subsequent engagement scenarios.

Interchange of Simulation Models and Algorithms
To complete a MSDE, aspects of the simulation models and environment must be incorporated to provide for METT-T assessment of the situation. Simulation algorithms can be integrated through the development of reliable interfaces (COM, JNI) between simulation assets and the MSDE. Data shared from the scenario file can be used as inputs to these interfaces. Most of the mission detail is provided through operational views of the scenario file. As each phase or scenario is completed, appropriate enemy information that has been collected for AAR can be shared to aid in the estimates of the enemy situation. This applies to the friendly situation as well. Statistical data representing the effectiveness of munitions against weapon systems and probabilities of hit and kill (Pk/Ph) can be shared from the simulation to help the commander determine how to best utilize resources under a given situational threat. Terrain analysis can be supported through the integration of a 3D stealth technology with the visual and/or correlated databases being used. The exchange of the scenario's environmental data (haze, rain, fog, time of day, etc.) would provide the details needed to accurately represent the environment visually. This data can also be used for map views of the scenario, including range fans and/or foils depicting intervisibility, engagement, and sensor coverage. Timing can be evaluated by providing estimates for executing various behaviors such as the travel time along a route.

CONCLUSION

The combination of today's standards for data representation, interoperability, and mechanisms for application integration provide the modeling and simulation community with the opportunity to integrate applications dealing with simulation, operational C4I systems, training development, and research, development & analysis seamlessly into the M&S user's desktop. In short, M&S technologies can be merged with the end user's applications of choice making those technologies part of their own in such a fashion as to enable a military scenario development environment to model the operational environment of today's military.

REFERENCES

Lacy, L.W.; MAJ Dugone T.D. (2001) Military Scenario Definition Language (MSDL) (SNAP Submission) U.S. Army TRADOC Analysis Center-Monterey

SEDRIS. (October 12, 2000) SEDRIS Technology Components http://www.sedris.com

W3 World Wide Web Consortium. (13 November 2000) Document Object Model (DOM) Level 2 Core Specification Version 1.0 W3C recommendation http://www.w3.org/TR/DOM-Level-2-Core

Norman Walsh (October 03, 1998) What is XML? O'REILLY XML.COM (online) http://www.xml.com/pub/a/98/10/guide0.html

G. Ken Holman (August 16, 2000) What is XLM? O'REILLY XML.COM (online) http://www.xml.com/pub/a/2000/08/holman/index.html

Norman Walsh (October 03, 1998) What is XLM? O'REILLY XML.COM (online) http://www.xml.com/pub/a/98/10/guide0.html

Eric van der Vlist (November 29, 2000) Using W3C XML Schema O'REILLY XML.COM (online) http://www.xml.com/pub/a/2000/11/29/schemas/part1.html

Tim Bray (January 24, 2001) What is RDF? O'REILLY XML.COM (online) http://www.xml.com/pub/a/2001/01/24/rdf.html

Jeffrey Richter (March 1996) http://msdn.microsoft.com/library/default.asp?URL=/library/periodic/period96/s28f.htm

FM 100-5 Operations (1993) Washington DC; Headquarters Department of the Army.

D. Kramer (March 15, 1997) Sun Microsystems. Java Native Interface Specification. Introduction Page. http://www.java.sun.com/products/jdk/1.1/docs/guide/jni/spec/intro.doc.html